クリーク分割問題の GA による効率的解法について

市瀬　和也・平林　永行・片山　謙吾*・成久　洋之**

岡山理科大学大学院工学研究科修士課程情報工学専攻

*岡山理科大学大学院工学研究科博士課程システム科学専攻

**岡山理科大学工学部情報工学科

(1997年10月6日　受理)

1．序　　論

グラフのクリーク分割問題（Clique Partitioning Problem：CPP）は，VLSI 回路の設計や並列処理システムの設計などに応用され，組み合わせ最適化問題の1つとして，NP 困難な問題とされている。遺伝的アルゴリズム（Genetic Argorithm：GA）は生物の進化の過程を模倣した比較的単純な基本的原理に基づいており，従来の手法では解決が困難であったさまざまな最適化・探索の問題に対して，許容される近似解を得ることができる有効な手法とされている。本研究では CPP の解法に GA を用い，Local Search と比較することでいかほどの優良な解が得られるかを検討するものである。

2．クリーク分割問題

2.1 クリーク

クリークとは，無向グラフ $G=(V, E)$ において節点集合 $W \subseteq V$ の生成部分グラフ $G(W)$ が完全部分グラフであるときに，W をクリークという。例を挙げると，図1の完全グラフの部分グラフであり，かつ完全グラフである図2はクリークである。

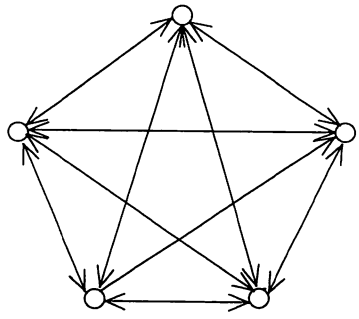

図1　完全グラフ

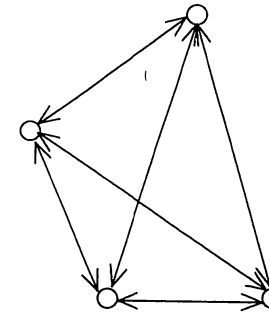

図2　図1の部分グラフ

2.2 クリーク分割問題

完全グラフ $G=(V, E)$ と，そのエッジ $(i, j)\in E$ に対応したコスト C_{ij} が与えられたとき，

$$Min \quad \sum_{1\leq i,j\leq j} c_{ij}p_{ij} \tag{1}$$

$$subject \quad to \quad p_{ij}=p_{ji} \quad \forall i, j\in V \tag{2}$$

$$p_{ii}=1 \quad \forall i\in V \tag{3}$$

$$p_{ij}+p_{jk}-p_{ki}\leqq 1 \quad \forall i, j\in V \tag{4}$$

$$p\in\{0, 1\} \quad \forall i, j\in V \tag{5}$$

のように式が定められ，コストを最小にする p_{ij} を求めることで望ましい分割を決定する。

3. 遺伝的アルゴリズム

3.1 遺伝的アルゴリズムについて

GA は生物の進化の過程にヒントを得た比較的単純な基本原理を基にしており，ほとんどの最適化・探索の問題に適用可能な枠組みである。一般的な GA は基本的な遺伝的操作として，選択淘汰，交差，突然変異と呼ばれる操作が存在する。GA は遺伝子をもつ仮想的な生物の集団を計算機内に設定し，あらかじめ定めた環境（条件）に適応している個体が子孫を残すように世代交代シミュレーションを実行することによって，遺伝子および生物集団を進化（適応）させていく。この生存競争に基づく確率的探査原理によって組み合わせ問題のような計算量の多い問題の近似解または最適解を比較的容易に短時間で得ることができるとされている。GA は実際のプログラミングの詳細を規定しない，緩やかな枠組みのため，各種の規則やパラメータの設定方法など，不確定要素が多い方法論である，と指摘されることがあるが，むしろ緩やかな枠組みであるために応用範囲が広いともいえる。

3.2 単純 GA のアルゴリズム

ここでは，本研究で用いた基本的な単純 GA のアルゴリズムについて述べる。

1．初期集団の発生

GA では探索空間中に複数の探索点，つまり複数の個体を設定してそれらの協調あるいは競合を行って探索を行う。探索開始時には探索空間はブラックボックスであり，どのような個体が望ましいかは不明である。このため，通常は初期の生物集団はランダムに発生させることになる。探索空間に対して何らかの予備知識がある場合は，評価値が高いと思われる部分に対して生物集団を発生させるなどの処理を行い，近似解の導出を高速にすることもある。

2．適応度の計算

生物集団中の各個体の環境との適応度を，あらかじめ決定されていた計算方法に基づいて計算する。計算方法は問題によってそれぞれ異なる。

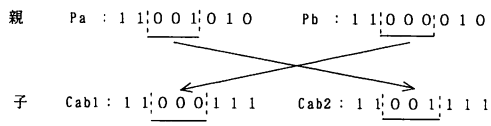

図3　2点交叉の例

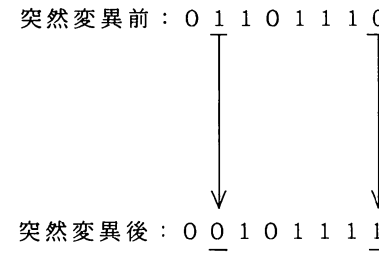

図4　突然変異の例

表1　3種類の問題

	Wildcats	UNO 1	UNO 2
頂点数	30	54	158
要素数	14	3	3
最適クリーク数	4	6	6

3．交叉の実行

生成されたN個の個体から2つの個体のペアを選択し，それぞれに対して交叉(crossover)と呼ばれる操作を行う。本研究で行った2点交叉を以下に説明する。

図3のように，2点交叉は2つの遺伝子型（親）に対してランダムに選んだ交叉位置を2点間で区切り，それぞれの区切られた場所を交換することで子孫を生成する。このため，子は6体できることになる。交叉によってできたこの個体は親の形質を継承した個体ということになる。この操作は探索空間上において新たな探索点を生成させることに相当している。初期の生物集団はさまざまな遺伝子を持つ個体が存在するため，交叉によってさらにバラエティに富んだ個体が生じる。世代が進むことで生物集団がある傾向に定まりつつある場合，それらの子も似通った遺伝子型になる。つまり，はじめは探索空間を大局的に探索し，傾向をつかむとより詳細に調べていくという操作が実現できることになる。

4．選択・淘汰の実行

本研究における生物集団の選択（selection）は，適応度の高いものを数体残し，残りの個体はすべて淘汰する（エリート保存戦略)。

5．突然変異の実行

生物集団が局所解におちいった場合でもそこからの脱出の可能性を持たせるため，突然変異（mutation）を行う。あらかじめ定めた生起確率である突然変異率によって突然変異を起こさせる。本研究では変異を起こした個体の1つの要素をランダムなクリークラベルに変化させるものとした。

6．終了条件

生成された生物集団が，進化シミュレーションを終了するかどうかの判定を行う。最適解や必要な精度の近似解がわかっている場合などは終了条件にそれを用いる場合がある。また，はじめに世代交代回数を決定しておき回数が満たされることで終了させる場合がある。本研究ではどの程度の近似解が得られるのかを調べるために回数を決定し，それによって終了することとした。終了条件が満たされていない場合は2の適応度の計算の処理に戻り，シミュレーションを続行する。

4．local Search

本研究では3種類の Local Search（LS）をつくり，GA と比較した。LS は山登り法（Hill clime：HC），確率的山登り法（SHC 1），改良確率的山登り法（SHC 2）を用いた。山登り法とは簡単なアルゴリズムによって，なるべく低い値（または高い値）を目指して探索する方法である。名前の通りに登山者で例えると，登山者は自分の周りしか見ることができない。そして，登山者は限られた時間内に自分の周りでなるべく標高の低い地点へと移動する。自分の周りが今の標高よりも低くない場合はその時点で探索を終了する。もちろん，この方法では必ずしも最も低い地点に到達できるとは限らない。問題の山にはいくつもの谷があるのが普通であるからである。この方法では谷（局所解）におちいった場合に脱出することができない。

次に，確率的山登り法は局所解におちいった場合にも脱出が可能なように，あらかじめ定めておいた確率によって GA の突然変異のようにランダムに探索点を変化させる。本研究ではクリーク分割をコード化した配列の要素を1つランダムに選択し，その値を変化させることによって実現している。

5．実験内容

5.1 実験問題

本研究ではベンチマーク問題で最適解がわかっている3種類の問題について GA と3種類の Local Search（LS）を用いて比較検討を行った。3種類の問題を以下にしめす（表1）。Wildcatsは野生のネコ科の動物の分類，UNO 1は国連参加国54ヶ国の分類，UNO 2は国連参加国158ヶ国の分類を扱った問題である。

Wildcats の分類は次のような特性で行っている。表2にまとめて示す。

1．毛の状態

1：斑点がなく1色　2：斑点がある　3：縞模様　4：白

2．毛の長さ

1：短い　2：長い

3．耳の形

1：丸い　2：尖っている

4．肩の高さ

1：50cm 以下　2：50～70cm 未満　3：70cm 以上

5．体　重

1：10kg 以下　2：10～80kg 未満　3：80kg 以上

6．体　長

1：80cm 以下　2：80～150cm 未満　3：150cm 以上

7．体長と尾の長さの比較

1：体よりも短い　2：同じ　3：尾のほうが長い

8．犬　歯

1：ほとんど発達していない　2：とても発達している

9．咽　頭

1：ない　2：ある

10．爪が出し入れできるか

1：できない　2：できる

11．活動する時間

1：昼　2：昼と夜　3：夜

12．獲物の大きさ

1：大きい　2：大小どちらでも　3：小さい

13．木に登るか

1：登らない　2：登る

14．獲物は待つか追うか

1：待つ　2：追う

表2の特性表からそれぞれについての特性ベクトルを導き出す。要素が同じ特性（同じ値）を持つ場合には1，持たない場合は0が特性ベクトルになる。例を挙げると，毛の状態の特性ベクトルは表3のようになる。

この特性ベクトルから，コストを計算する。望ましい分割を決定しうる変数を P，特性を表す変数を R_k とおくと，この問題は次のように定式化できる。

$$Min\sum_{k=1}^{q}\delta(P, R_k)$$

ここで q は特性の数を表す。同値関係である P および R_k はそれぞれ特性ベクトル p_{ij} および r_{ij}^k で表すことができるので

$$\delta(P, R_k) = \sum_{i\leq i, j\leq n}(p_{ij}-r_{ij}^k)-2$$

となり，

$$Min \sum_{k=1}^{q} \sum_{i \leq i,j \leq n} (P_{ij} - r_{ij}^{k})^2$$

となる。ただし，p_{ij}^{k} および r_{ij}^{k} は式(2)(3)(4)(5)が条件となる。さらにこれを簡略化すると，

$$Min \quad Cst + \sum_{k=1}^{q} \sum_{i \leq i,j \leq n} p_{ij}(1 - 2r_{ij}^{k})$$

となり，

$$Min \quad Cst + \sum_{i \leq i,j \leq n} p_{ij}(q - \sum_{k=1}^{q} 2r_{ij}^{k}) \tag{6}$$

となる。よってコスト c_{ij} は

表2 Wildcats の特性

Wildcats	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1 Lion	1	0	1	3	3	3	2	1	1	1	1	1	0	1
2 Tiger	3	0	1	3	3	3	2	1	1	1	3	1	0	0
3 Jaguar	2	0	1	3	3	2	1	1	1	1	2	1	1	0
4 Leopard	2	0	1	3	3	2	2	1	1	1	3	2	1	0
5 Once	2	1	1	2	2	2	3	1	1	1	1	2	1	0
6 Guepard	2	0	1	3	2	2	3	0	0	0	1	2	0	1
7 Puma	1	0	1	2	3	2	3	1	0	1	2	2	1	0
8 Panth. Nebuleuse	4	0	1	2	2	2	3	1	1	1	3	3	1	0
9 Serval	2	0	2	2	2	2	1	0	0	1	1	3	1	1
10 Ocelot	2	0	1	2	2	2	2	0	0	1	2	3	1	0
11 Lynx	2	1	2	2	2	2	1	1	0	1	2	2	1	0
12 Caracal	1	0	2	2	2	1	1	0	0	1	2	3	1	1
13 C. Viverrin	2	0	1	1	1	2	2	0	0	1	2	3	0	0
14 Jaguarundi	1	0	1	1	2	2	3	0	0	1	3	3	1	0
15 C. Chaus	1	1	2	1	2	1	2	0	0	1	3	3	1	0
16 C. Dore	1	0	1	1	1	1	2	0	0	1	3	3	1	0
17 C. Marguay	2	0	1	1	1	1	2	0	0	1	3	3	1	0
18 C. Margerite	1	1	1	1	1	1	2	0	0	1	2	3	0	0
19 C. Cafer	3	0	1	1	1	1	2	0	0	1	3	3	1	1
20 C. Chine	1	0	2	1	1	1	1	0	0	1	2	3	1	0
21 C. Bengal	2	0	1	1	1	1	2	0	0	1	3	3	1	0
22 C. Rouilleux	2	0	1	1	1	1	2	0	0	1	2	3	1	0
23 C. Malais	1	1	1	1	1	1	1	0	0	1	3	3	1	0
24 C. Borneo	1	0	1	1	1	1	2	0	0	1	3	3	1	0
25 C. Negripes	2	0	1	1	1	1	1	0	0	1	2	3	1	1
26 C. Manul	1	1	1	1	1	1	1	0	0	1	3	3	1	0
27 C. Marble	4	0	1	1	1	1	3	0	0	1	3	3	1	0
28 C. Tigrin	2	0	1	1	1	1	2	0	0	1	3	3	1	0
29 C. Temminck	1	0	1	1	1	1	2	0	0	1	3	3	1	0
30 C. Andes	2	1	1	1	1	2	2	0	0	1	3	2	1	0

表3　毛の状態の特性ベクトル

Wildcats	1	2	3	4	5	6	7	8	9	10	…	29	30
1	1	0	0	0	0	0	1	0	0	0	…	1	0
2	0	1	0	0	0	0	0	0	0	0	…	0	0
3	0	0	1	1	1	1	0	0	1	1	…	0	1
4	0	0	1	1	1	1	0	0	1	1	…	0	1
5	0	0	1	1	1	1	0	0	1	1	…	0	1
6	0	0	1	1	1	1	0	0	1	1	…	0	1
7	1	0	0	0	0	0	1	0	0	0	…	1	0
8	0	0	0	0	0	0	0	1	0	0	…	0	0
9	0	0	1	1	1	1	0	0	1	1	…	0	1
10	0	0	1	1	1	1	0	0	1	1	…	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
29	1	0	0	0	0	0	1	0	0	0	…	1	0
30	0	0	1	1	1	1	0	0	1	1	…	0	1

$$\begin{cases} c_{ij} = 0 \\ c_{ij} = q - 2\sum_{k=1}^{q}\gamma - k_{ij} \qquad \forall i, j : i \neq j \end{cases} \tag{7}$$

となり，その結果，式(6)は

$$Min \sum_{i \leq i, j \leq n} c_{ij} p_{ij}$$

となる。ただし，条件は式(2)(3)(4)(5)である。この式から，この問題は CPP と等価であるといえる。そこで，式(7)に特性ベクトルを代入することでコストが求められる。

また，UNO 1 と UNO 2 の問題も同様のため省略する。

5.2 パラメータの設定

GA のパラメータ設定は次のように設定する。

個体数　　：10

交叉確率　：1.0

突然変異率：0.05

打ち切り世代数は Wildcats，UNO 1 は1000，UNO 2 は4000とする。HC では局所解が算出されるまで計算を行った。SHC 1，2 での打ち切り世代数は Wildcats, UNO 1 で1000, UNO 2 で150とした。また，確率は Wildcats，UNO 1 を0.005，UNO 2 を0.001として計算を行った。

それぞれのアルゴリズムについて10回の試行を行い，平均世代数，適応度(最良，最悪，平均)，平均計算時間，平均クリーク数を計算した。本実験では Intel 社の MMX 搭載 Pentium 200 MHz を使用して計算を実行した。

5.3 実験結果

本研究では，GA と各種の近似解法を用いてその性能を比較した。各表からわかるよう

に，頂点数が小さな場合には LS によっても比較的良好な解が得られている。しかし，頂点数が増えるごとに計算時間が増大し，また，適応度も悪くなる。これに対して GA では安定して良い適応度を得ることができ，組み合わせ爆発を起こす大きな問題に対して比較的高速に良好な解が得られていることがわかる。ただし，最も高い適応度を与える分割を求めるために，良い適応度はクリーク分割数を小さくするとは限らない。

参考文献

1) D. Snyers, "Clique partitioning problrm genetic algorithms", Artificial neural nets and genetic algorithms, pp.352-360, 1993.

2) M. Grötschel, Y. Wakabayasi "A cutting plane algorithm for a clustering probrem", Mathematical Programming **45**,pp.59-96, 1989.

表 4 Wildcats における算出結果

	LS	SHC 1	SHC 2	GA
平均算出世代	6.5	148.5	511.5	78.8
最良適応度	−2784	−2844	−2936	−3028
平均適応度	−240.6	−2634.4	−2918.8	−3028
最悪適応度	−1904	−2396	−2908	−3028
平均計算時間(s)	0.472	4.042	16.242	1.300
平均クリーク数	13.9	7.2	8.3	4

表 5 UNO 1 における算出結果

	LS	SHC 1	SHC 2	GA
平均算出世代	1.9	424.3	423.9	468.6
最良適応度	−340	−702	−704	−1716
平均適応度	−246.4	−360.2	−620.8	−1641.6
最悪適応度	−210	−258	−566	−1530
平均計算時間(s)	0.890	65.109	87.878	33.816
平均クリーク数	28.5	14.7	9.6	6.4

表 6 UNO 2 における算出結果

	LS	SHC 1	SHC 2	GA
平均算出世代	1.8	32.6	43.5	2802.2
最良適応度	−588	−580	−588	−24868
平均適応度	−434.0	−459.0	−452.9	−24868
最悪適応度	−138	−136	−110	−24868
平均計算時間(s)	58.956	527.428	1314.319	432.414
平均クリーク数	78.1	66.9	61.0	6

Some Comments on Efficient Solving Method by using Genetic Algorithms for Clique Partitioning Probrem

Kazuya ICHISE, Hisayuki HIRABAYASHI,
Kengo KATAYAMA and Hiroyuki NARIHISA*
Graduate School of Engineering
**Department of Information & Computer Engineering*
Faculty of Engineering
Okayama University of Science
Ridai-cho 1-1, Okayama 700-0005, Japan
(Received October 6, 1997)

Clique Partitioning Probem (CPP) is classified as NP-hard optimization probrem. Clique partitioning probrem is one of convinational optimization probrem of graph. Therefore, clique partitioning probrem is considered to be NP-hard optimization probrem. This clique partitioning probrem has been used in wide range of application from VLSI circuit design to mapping parallel programs on parallel architectures.

Recently, Genetic Algorithms (GA) has been focussed attention to be strong algorithm which can solve a convinational optimization probrem in view point of practical application.

In this paper we represent the efficientry of genetic algorithms for solving the clique partitioning probrem.